
# Self-Adaptive Software with Model-Checking

Chungbuk National University
Department of Computer Science

Prof. Euijong Lee

# Personal History

**2006.2**
Graduated
Jeju Jeil High School

**2012.2**
B.S
Computer & Informatics
Korea University

**2018.8**
Ph.D.
Computer Science & Engineering
Korea University

**2020.9 ~ present**
Assistant Professor
Computer Science
Chungbuk National University

Now

**2012.8 ~ 2014.4**
Software Developer
- AtEar Project
(Wireless Network Security Tool)

**2017.9 ~2020.8**
Adjunct Professor
- Data Structure
- Software Engineering

**2018.9 ~2020.8**
Postdoctoral Researcher
Computer and Information Security
Sejong University

## Prof. Euijong Lee

Self-Adaptive Software | Internet of Things | Software Modeling

▷ Self-Adaptive Software Framework
▷ Self-Adaptive IoT Framework
▷ IoT with Machine Learning
▷ Big data for Recommendation System

# Relationship with KCSE

❖ KCSE 제 2회 소프트웨어공학 단기전문가 강좌

**한국정보과학회 소프트웨어공학 소사이어티**

**제 2 회 소프트웨어공학 단기전문가강좌**

**2013 년 7 월 22 일 – 24 일, 서강대학교**

**행사개요**

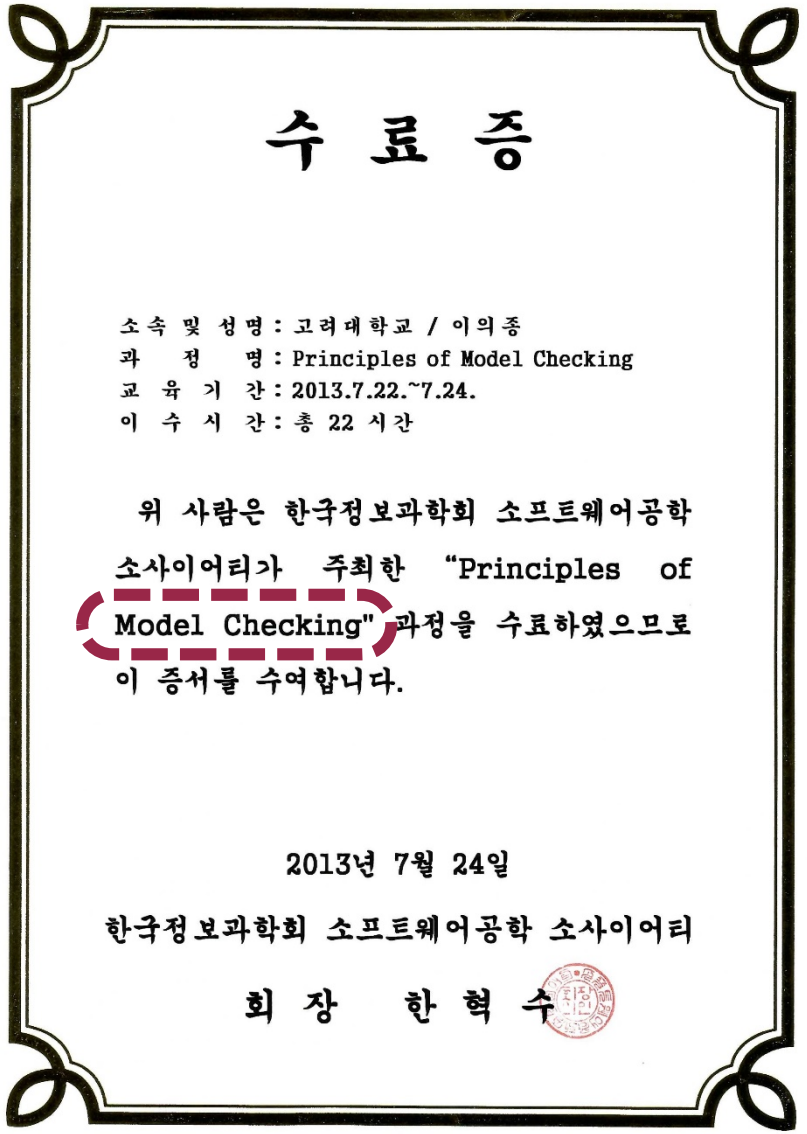
주제: Principles of Model Checking

일시: 2013.07.22(월) – 24(수) (3일간/20시간)

장소: 서강대학교 바오로관 203호

주최: 한국정보과학회

주관: 한국정보과학회 소프트웨어공학 소사이어티



수 료 증

소속 및 성명 : 고려대학교 / 이의종
과 정 명 : Principles of Model Checking
교 육 기 간 : 2013.7.22.~7.24.
이 수 시 간 : 총 22 시간

위 사람은 한국정보과학회 소프트웨어공학 소사이어티가 주최한 "Principles of Model Checking" 과정을 수료하였으므로 이 증서를 수여합니다.

2013년 7월 24일

한국정보과학회 소프트웨어공학 소사이어티

회 장 한 혁 수

# Contents


- Research Background
- RINGA: Self-Adaptive Framework
- RINGA-IoT: Self-Adaptive Framework for IoT
- Cache-based Model Abstraction
- On going & Future Research

# Research Background

# Framework

## ❖ What is framework?

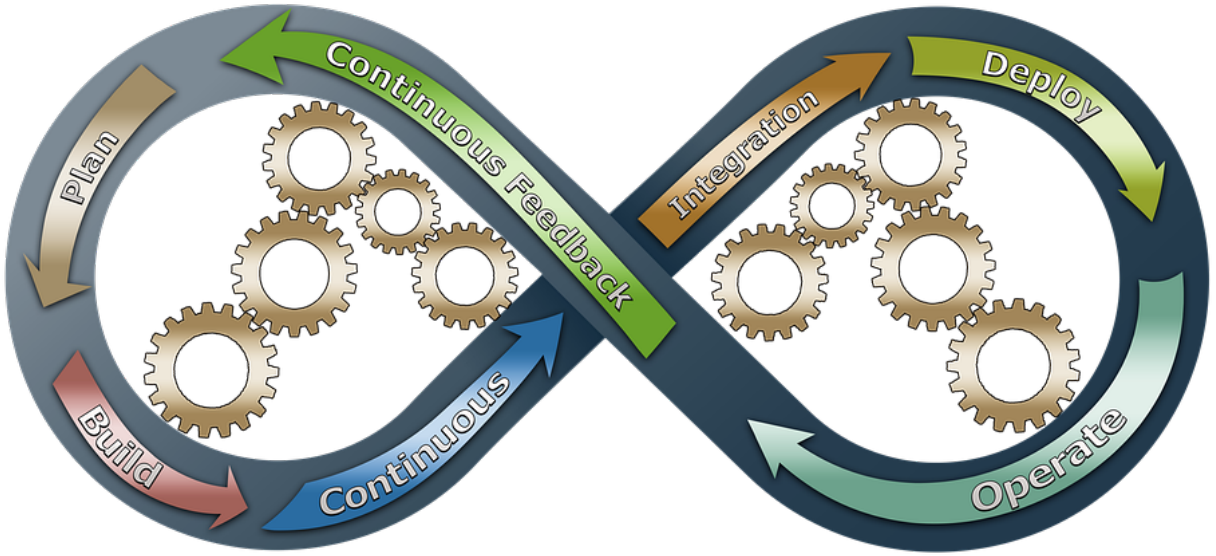
- A system of rules, ideas, or beliefs that is used to plan or decide something

Cambridge Dictionary

## ❖ 프레임워크 = 체계

- 일정한 원리에 따라서 낱낱의 부분이
  짜임새 있게 조직되어 통일된 전체

네이버 사전



# Self-Adaptive Software

❖ Self-Adaptive Software

"Self Adaptive Software **evaluates its own behavior and changes behavior** when the evaluation indicates that it is not accomplishing what the **software is intended to do, or when better functionality or performance is possible**"

DARPA 1997

❖ Self-Adaptive Software Environment

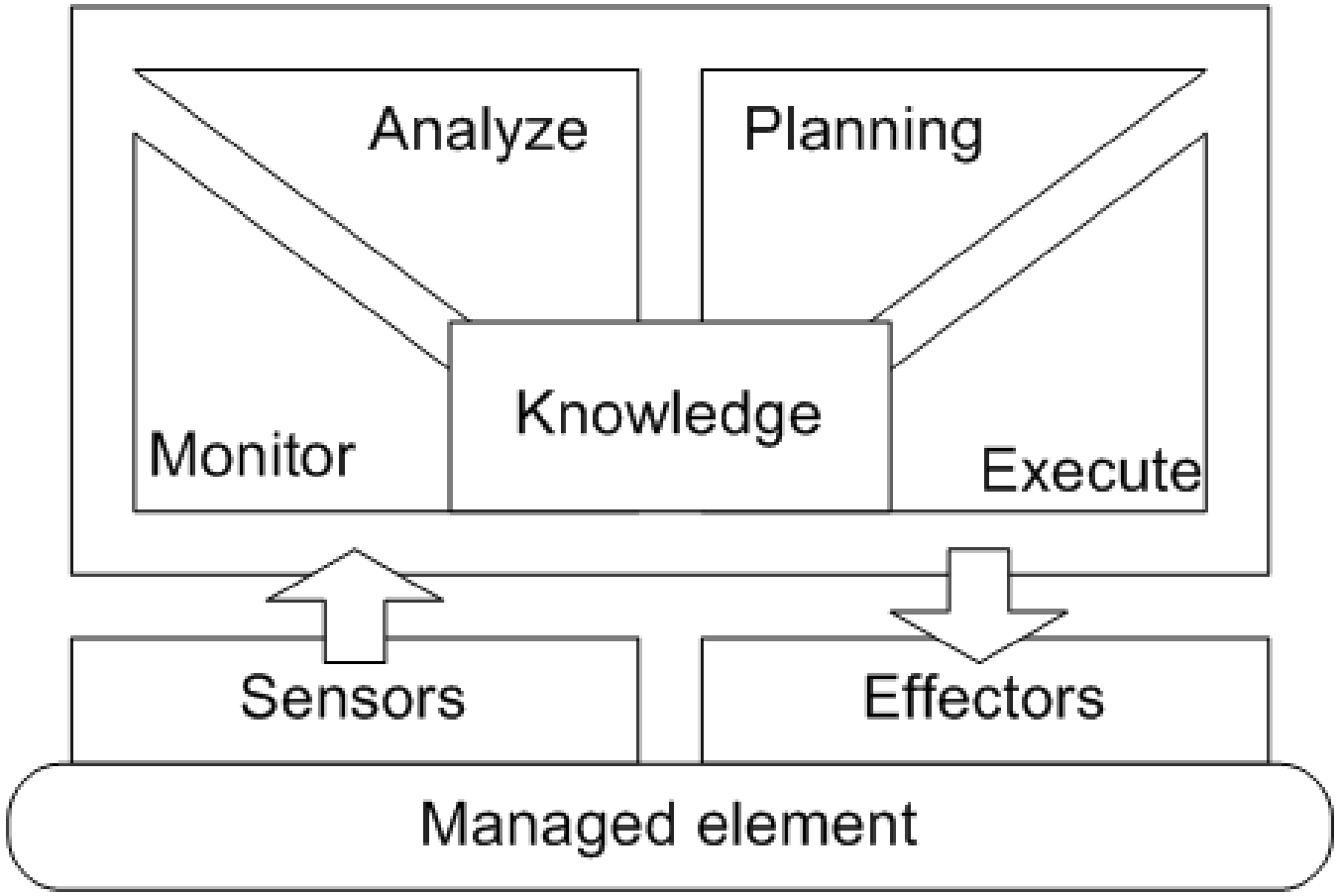
- Software needs to be applied to various environment (e.g., "Internet of Things," "the cloud," etc.)
- Software needs to adapt to dynamic environmental change

# Self-Adaptive Software

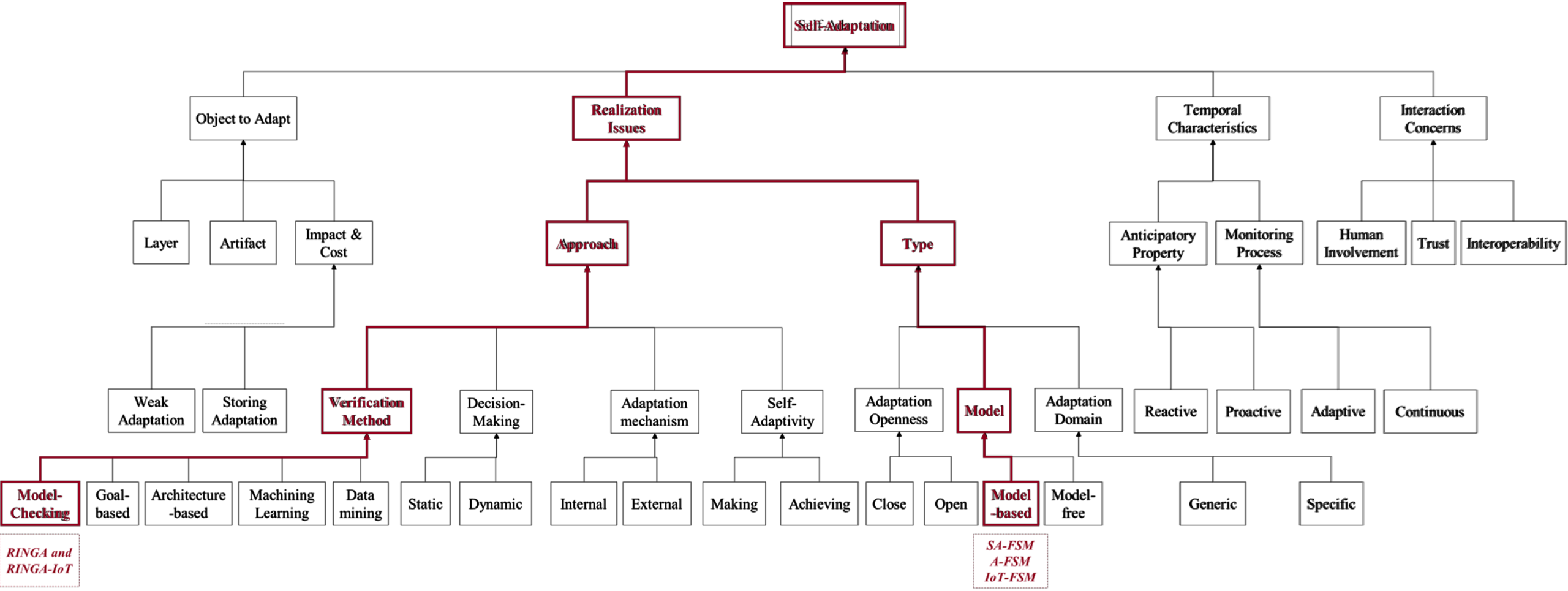
❖MAPE-K loop

- MAPE-K feedback loop is most influential reference control model for autonomic and self-adaptive system

- Monitoring, Analysis, Planning, Execution with Knowledge



# Self-Adaptive Software




Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. ACM transactions on autonomous and adaptive systems (TAAS), 4(2):14, 2009.

# Model-Checking

❖Exhaustive testing (complete testing)

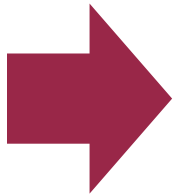
- All possible combinations are used for testing
- **Ideal, but impossible**

❖Example

- Integer variable is 32bit
- All possible combination $2^{32}*2^{32} = 2^{64}$

```
public void module(){
    int x;
    int y;

    //...
}
```

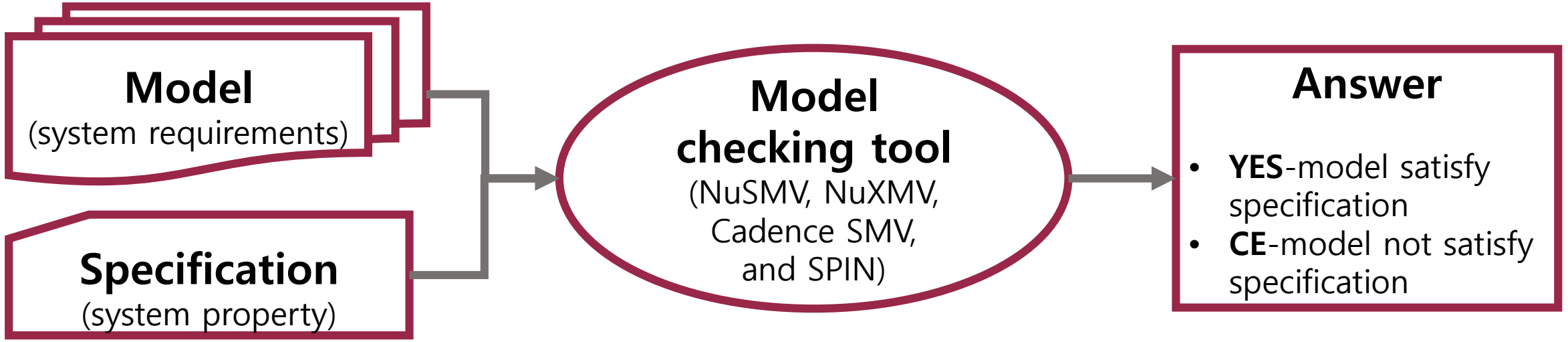


2^64 = 18,446,744,073,709,600,000
1 sec = 1,000,000
1 day = 86,400,000,000
1 year = 31,536,000,000,000
**Testcase / Year = 584942.4**

# Model-Checking

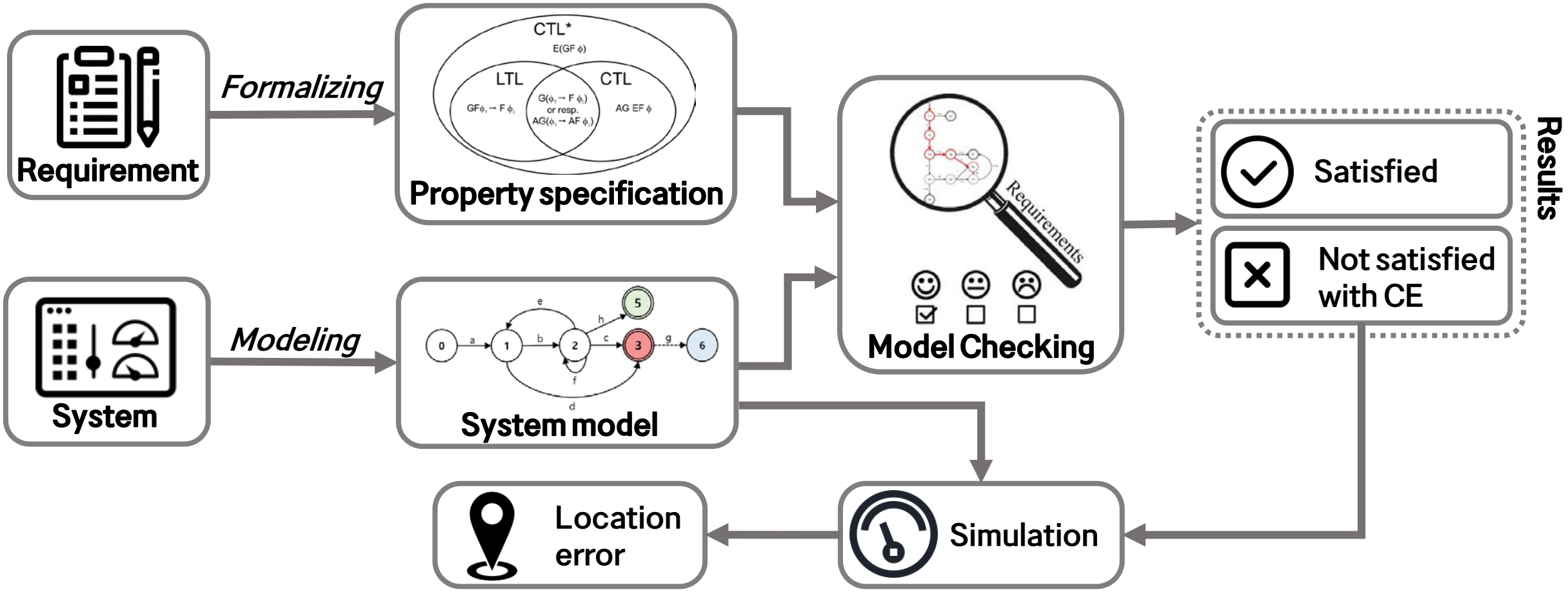
## ❖Model-Checking

- Model checking is one of the effective static verification methods for software and hardware
- Reduction, simplification, and abstraction
- Model is generated transition model (i.e., finite state machine)
- Specification (requirement) is presented as specific method (e.g., CTL and LTL)
- Results can be YES or CE (Counter Example)



# Model-Checking

❖Model-Checking



# Model-Checking

❖ Advantage
- Automation
- Counter example
- Without user involvement
- Verifying non-deterministic
- Can support various situations

❖ Disadvantage
- State explosion problem
- Gap between model between real system

How model-checking can be applied self-adaptive software at runtime?

# RINGA: Self-Adaptive Framework

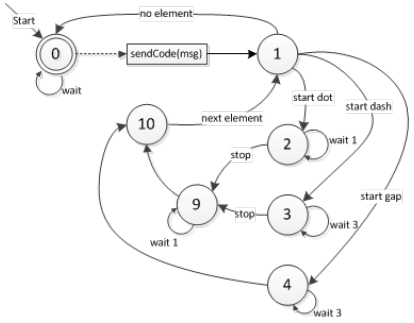
ACM/SIGAPP Symposium on Applied Computing/ SAC 2017
Information and software technology (2018)

# Objective

## ❖ Self-Adaptive Software Demand

- Research that integrates traditional verification and theory with self-adaptive software is in demand
  - **Model-checking** is one of the effective static methods for software
    - ✓ Given a model of a system, exhaustively and automatically check whether this **model** meets a given **specification**

### Finite State Machine




### LTL(Linear Temporal Logic)

Linear Time Temporal Logic (LTL) Semantics

Given an execution path x and LTL properties p and q

| | | |
|---|---|---|
| $x \models p$ | iff | $L(x_0, p)$ =True, where $p \in AP$ |
| $x \models \neg p$ | iff | not $x \models p$ |
| $x \models p \wedge q$ | iff | $x \models p$ and $x \models q$ |
| $x \models p \vee q$ | iff | $x \models p$ or $x \models q$ |
| $x \models X\, p$ | iff | $x^1 \models p$ |
| $x \models G\, p$ | iff | for all i, $x^i \models p$ |
| $x \models F\, p$ | iff | there exists an i such that $x^i \models p$ |
| $x \models p\, U\, q$ | iff | there exists an i such that $x^i \models q$ and for all $j < i$, $x^j \models p$ |

### CTL(Computational Tree Logic)

CTL Semantics

Given a state s and CTL properties p and q

| | | |
|---|---|---|
| $s \models p$ | iff | $L(s, p)$ = True, where $p \in AP$ |
| $s \models \neg p$ | iff | not $s \models p$ |
| $s \models p \wedge q$ | iff | $s \models p$ and $s \models q$ |
| $s \models p \vee q$ | iff | $s \models p$ or $s \models q$ |
| $s_0 \models EX\, p$ | iff | there exists a path $s_0, s_1, s_2, \ldots$ such that $s_1 \models p$ |
| $s_0 \models AX\, p$ | iff | for all paths $s_0, s_1, s_2, \ldots, s_1 \models p$ |

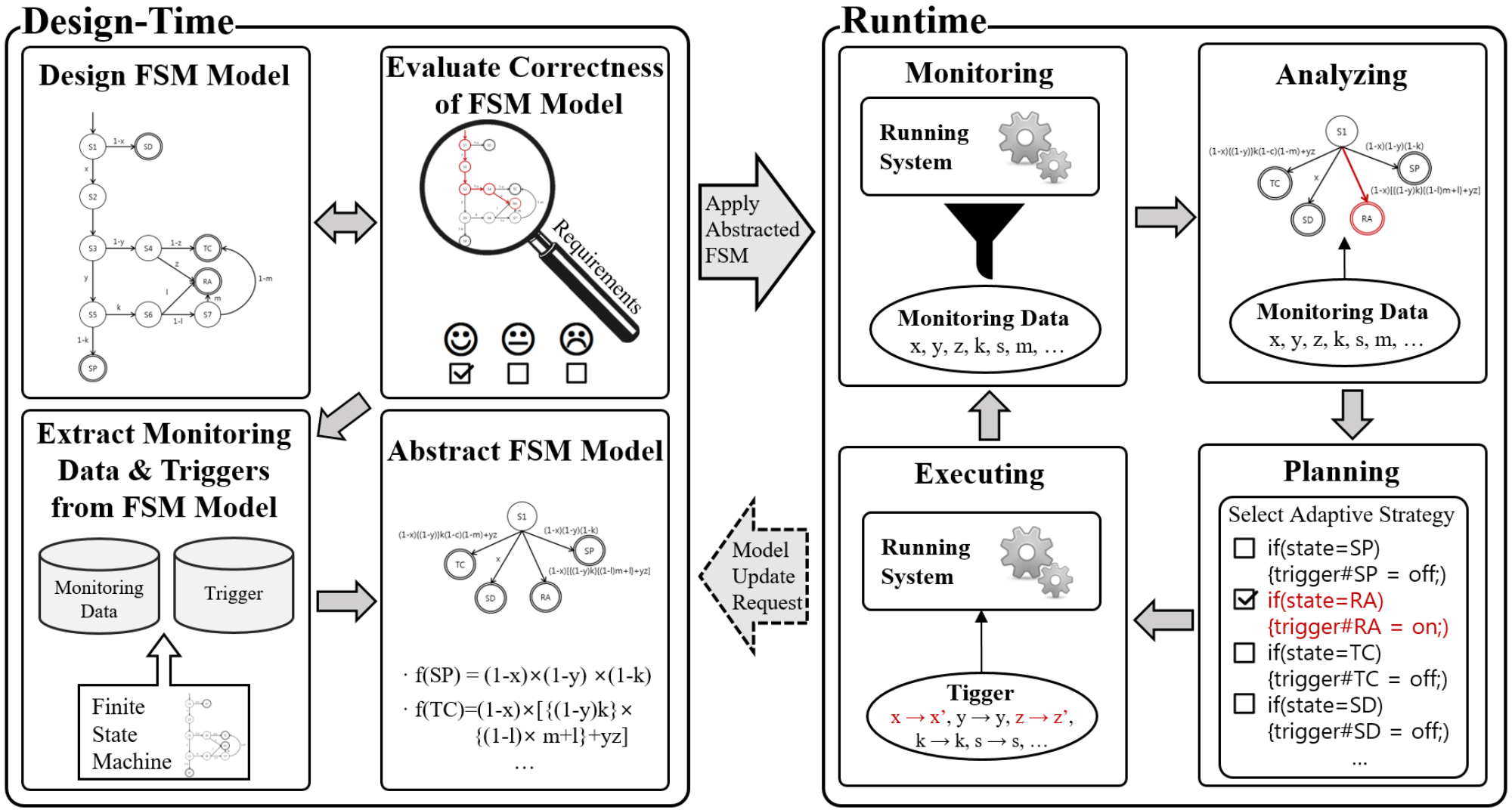
# Objective

❖ Objective : Self-Adaptive Software Verification with Model-Checking

- Model checking is one of the effective static verification methods for software
- Chronic problems(i.e., state explosion) need to be resolved at runtime

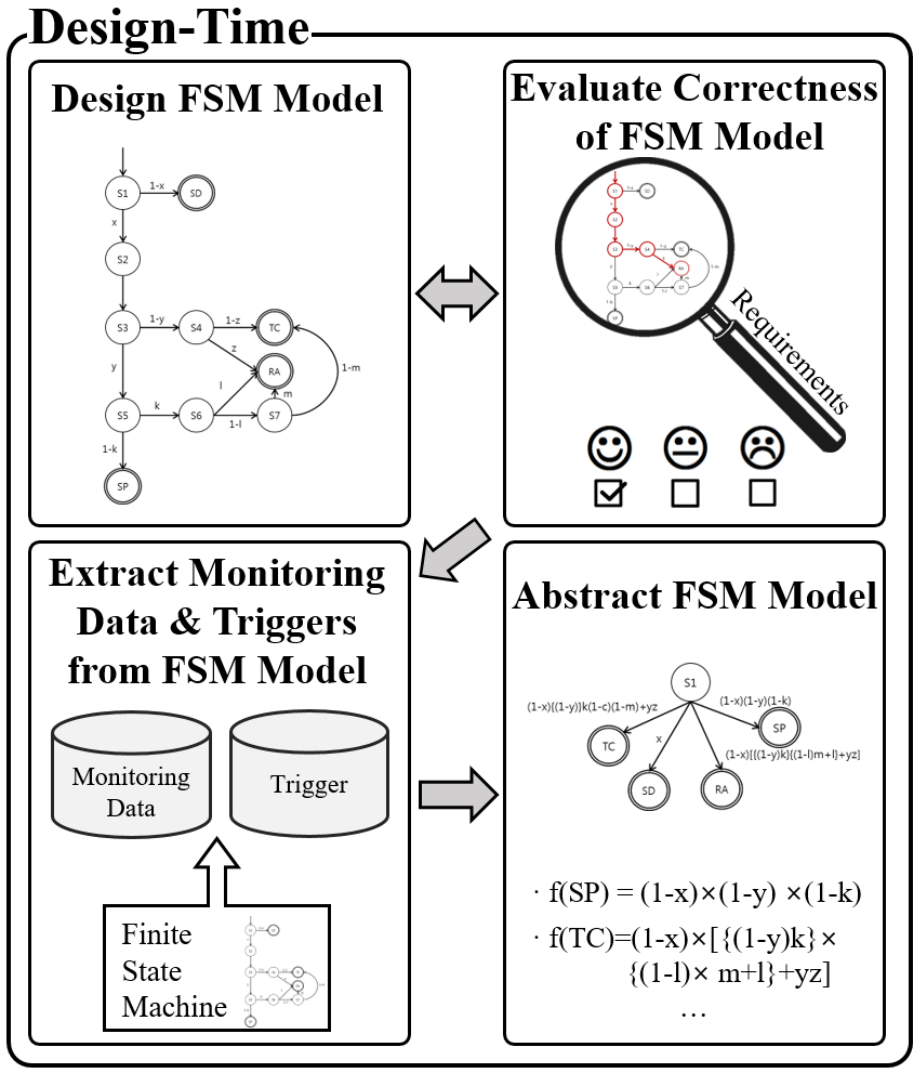
**A self-adaptive software framework is proposed that applies model checking for the software to verify itself at runtime (RINGA)**

# RINGA: Self-Adaptive Software Framework



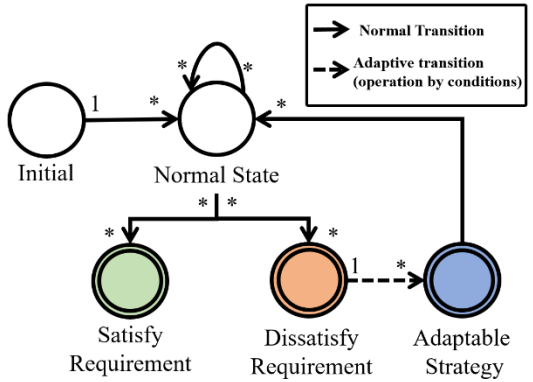

※ The abstraction algorithm is executed once

# RINGA: Self-Adaptive Software Framework

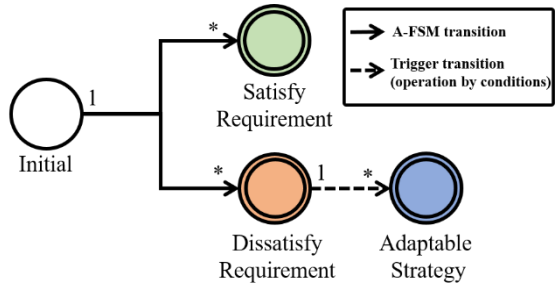



## ❖ Self-Adaptive Software Modeling

- SA-FSM (Self-Adaptive Finite State Machine)



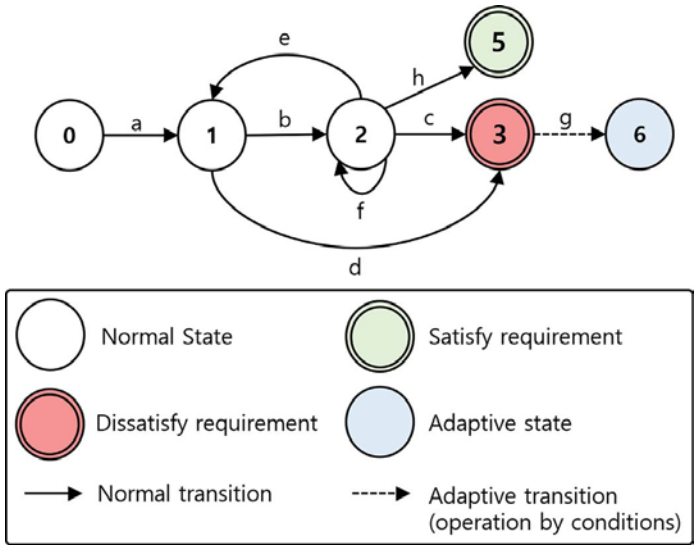

- A-FSM (Abstracted Finite State Machine)

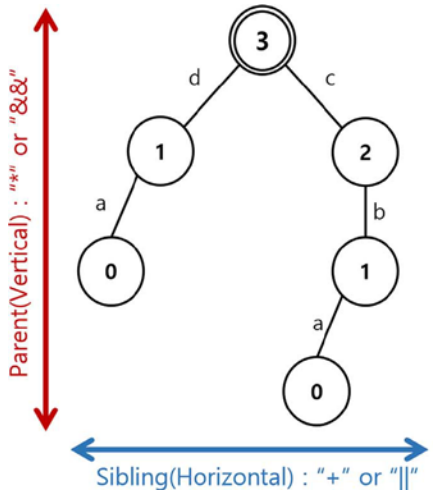


# RINGA: Self-Adaptive Software Framework

## ❖ Abstracting Processes

① SA-FSM Example ② Extracts Reachable paths to specific models ③ Conversion SA-FSM to A-FSM



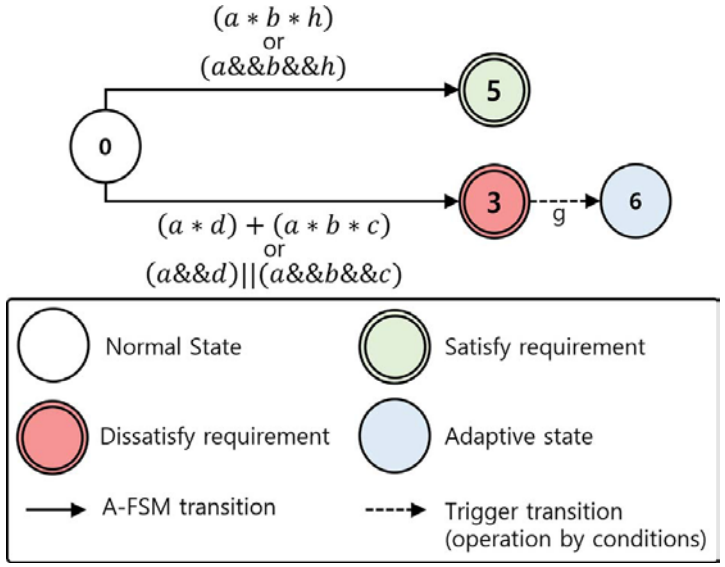




$f(3) = (a * d) + (a * b * c)$

Converted with "*" and "+"

Converted with "&&" and "||"

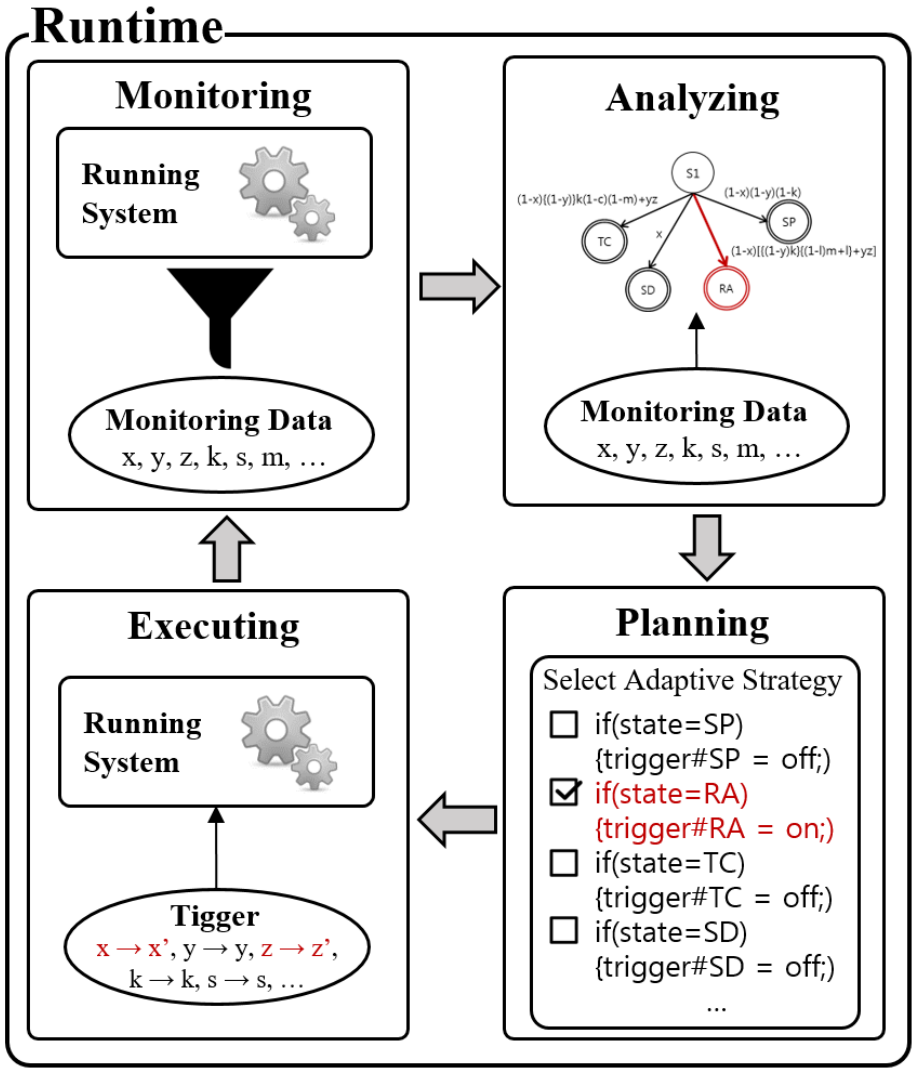
$f(3) = (a\&\&d)||(a\&\&b\&\&c)$



# RINGA: Self-Adaptive Software Framework

❖ Runtime with MAPE-loop

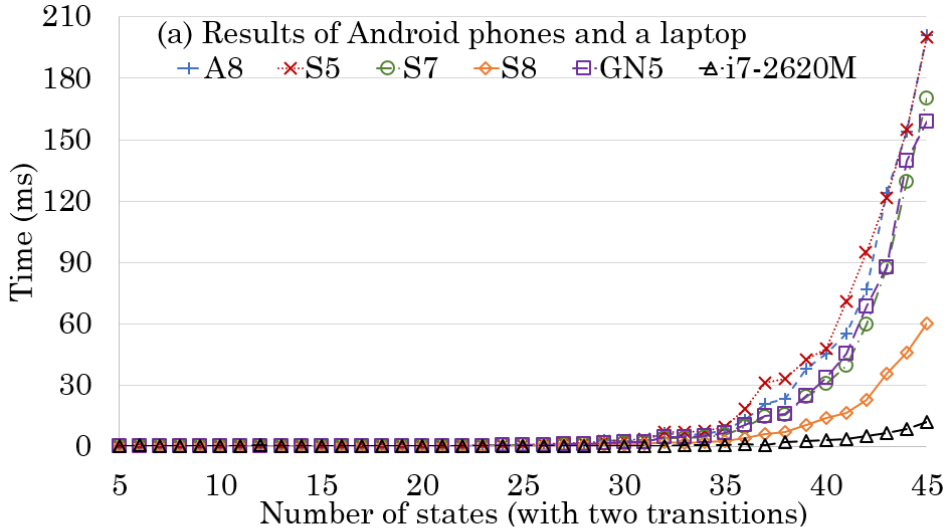
- **Monitoring**: collects data from the environment and internal software changes
- **Analyzing**: analyzes the symptoms related to adaptation situations using the monitored data (i.e., calculate A-FSM)
- **Planning**: triggers an adaptive strategy when adaptation is required
- **Executing**: activating the adaptive strategies

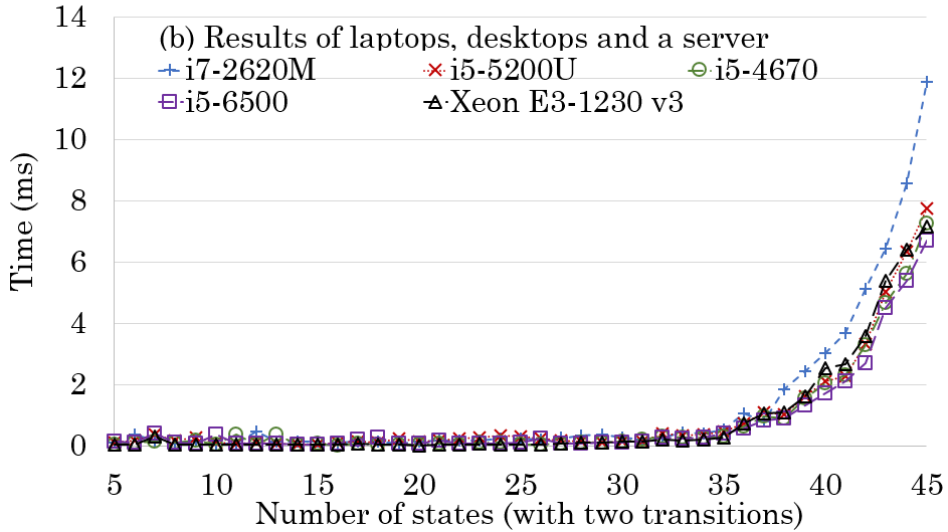


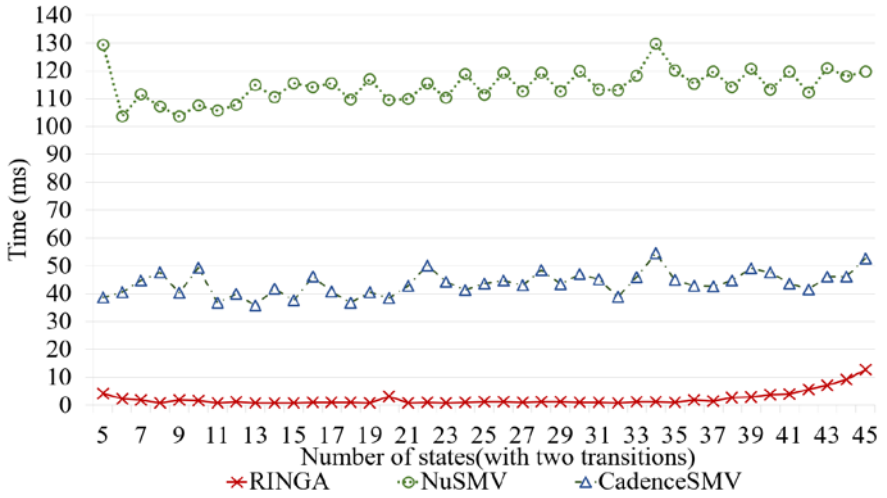
# RINGA: Self-Adaptive Software Framework

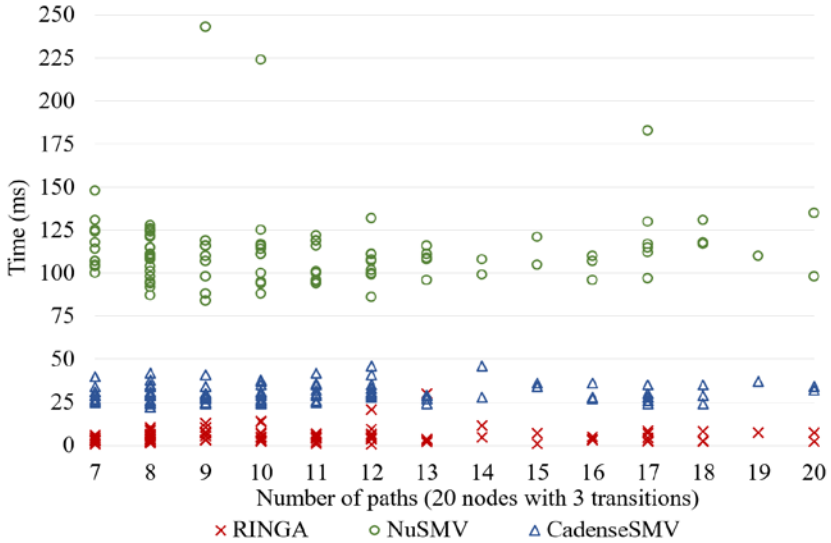
## ❖ Experimental Evaluation

- RINGA can be applied in various hardware & RINGA is efficient at runtime



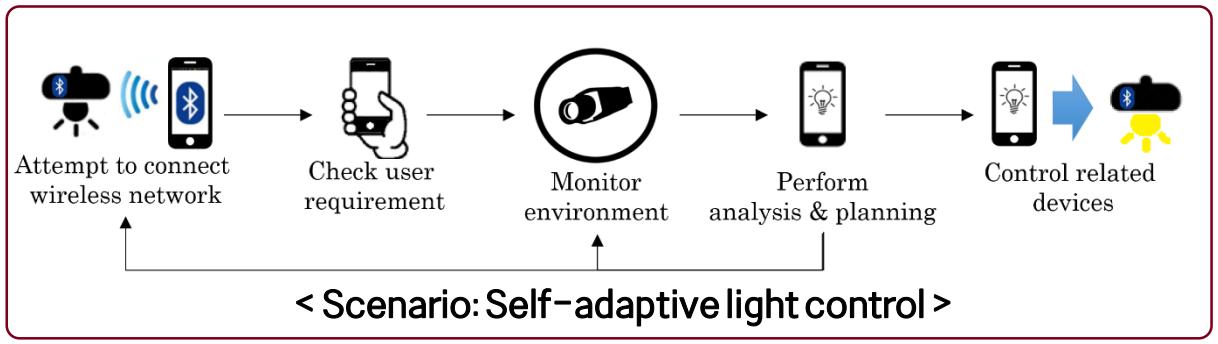



# RINGA: Self-Adaptive Software Framework

## ❖ Proof of Concept with IoT Light Control Scenario

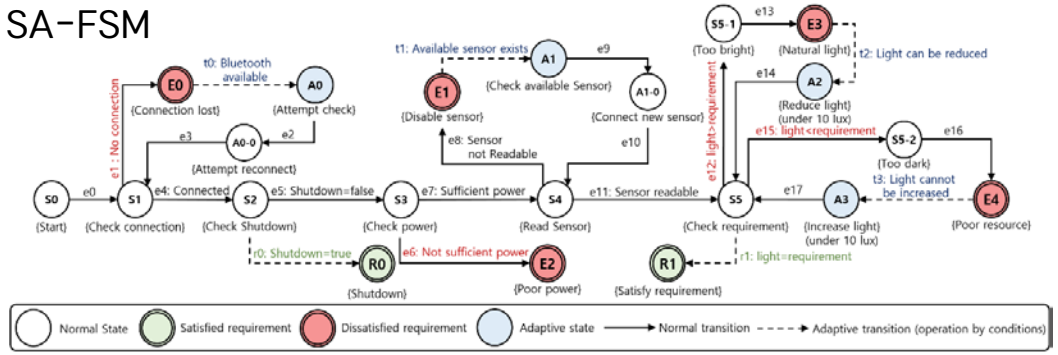
- RINGA is implemented with Java
- Android application and Arduino light controller are implemented
- RINGA performs reasonably well with adaptiveness to various environmental changes



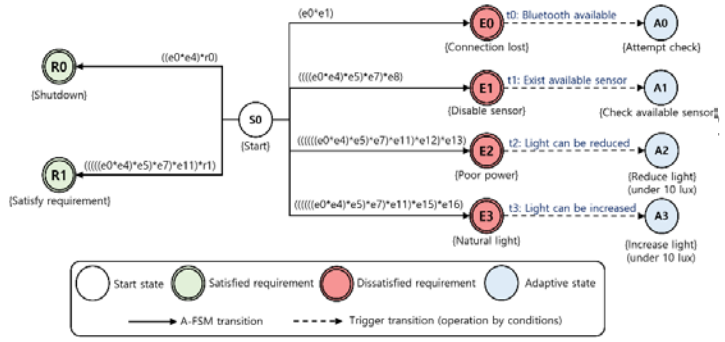

< Scenario: Self-adaptive light control >

< Implementation with Android and Arduino >

SA-FSM




A-FSM




< Modeling Result for the scenario>

# RINGA-IoT:
# Self-Adaptive Framework for IoT

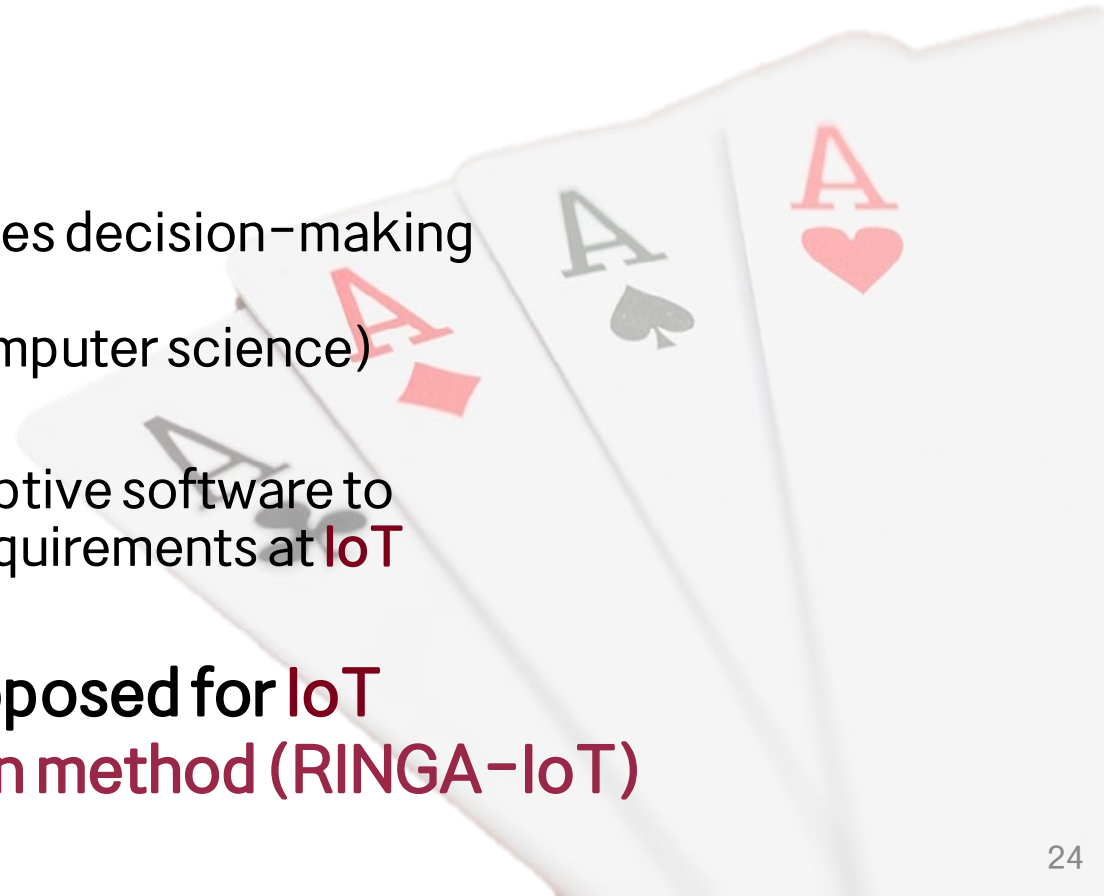
Sensors (2019)
Ambient intelligence & humanized computing (2020)

# Objective

❖ Objective: IoT System Modelling &
Strategy Extraction with Game Theory for IoT

- **IoT system modelling based on RINGA**
- **Game theory** is a mathematical theory that facilitates decision-making in a set of different stakeholders
(e.g., economics, political science, biology, and computer science)
- A game-theoretic method can be used in self-adaptive software to determine the optimal decisions under different requirements at **IoT**

**A self-adaptive software framework is proposed for IoT using a game-theoretic strategy extraction method (RINGA-IoT)**



# Simple Research Background

❖ Game Theory:

- **Game theory** is a mathematical theory that facilitates decision-making in a set of different stakeholders (e.g., economics, political science, biology, and computer science)

- The Nash equilibrium is one of the **foundational concepts in game theory**

# Simple Research Background

❖ Game Theory: Nash Equilibrium (simple example)

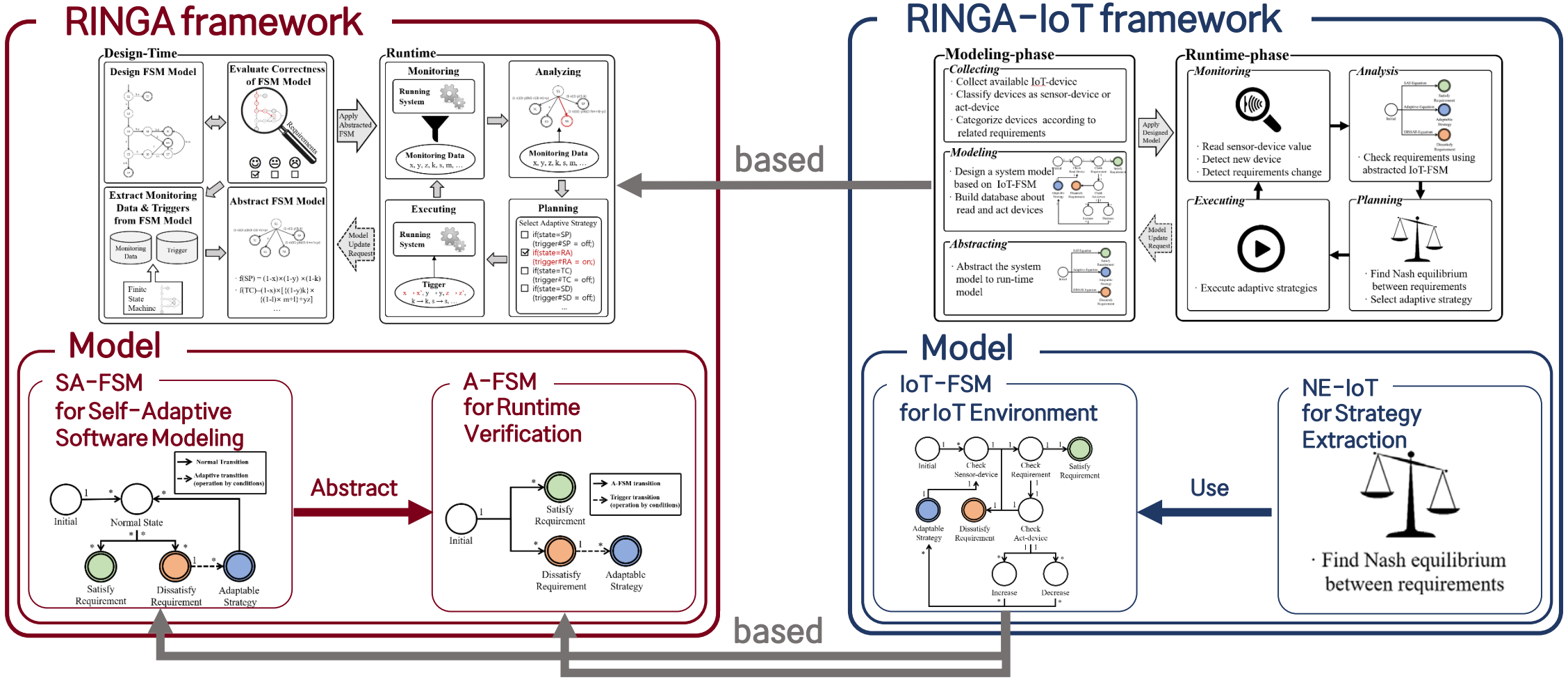
- Battle of the sexes game

| Benefit of man and woman | | Woman | |
|---|---|---|---|
| | | art gallery | football |
| Man | art gallery | (1, 3) | (0,0) |
| | football | (0,0) | (3, 1) |

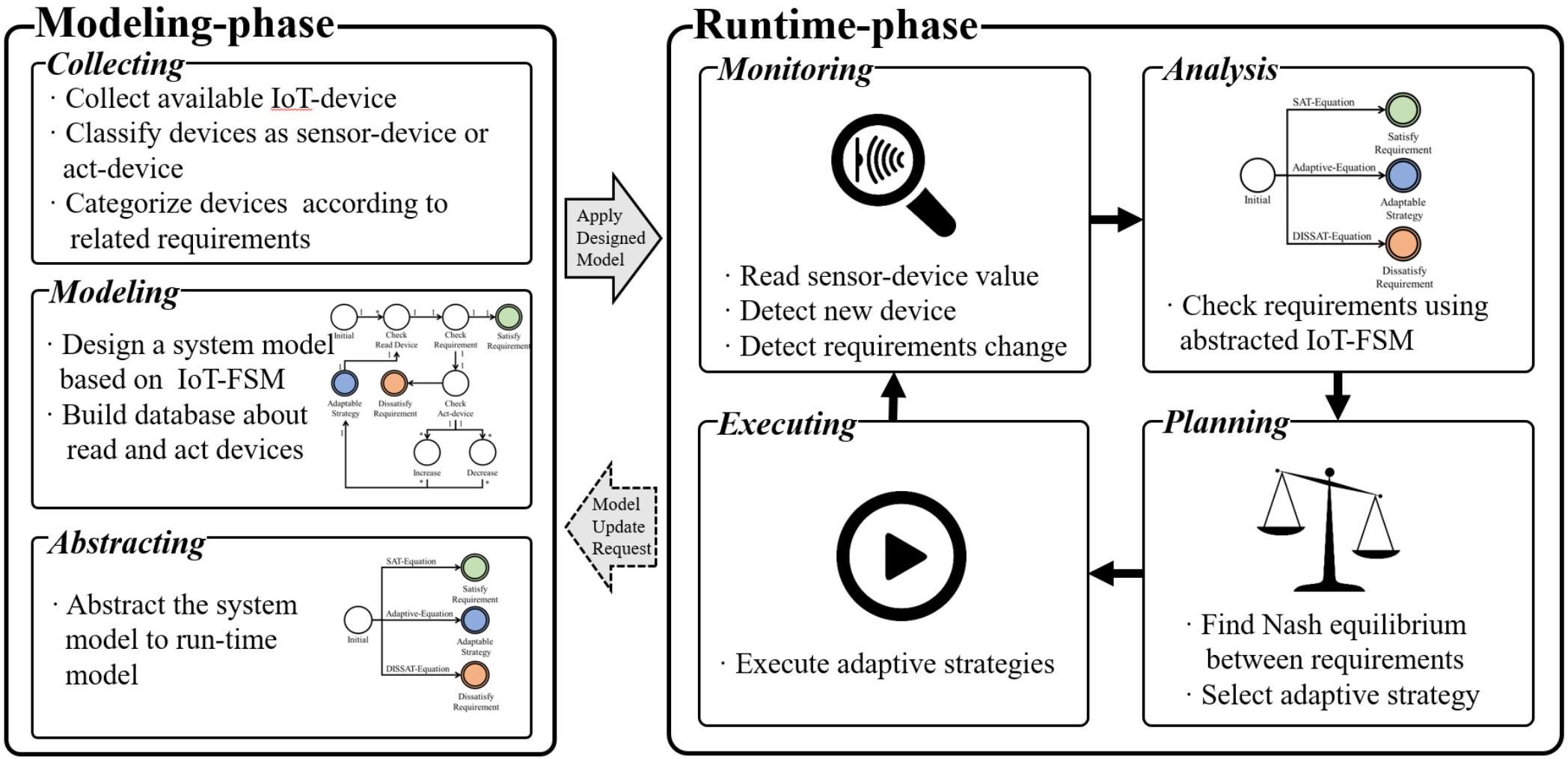
There are two Nash equilibrium,
and it is needed strategy rule to select a strategy

# RINGA-IoT: Self-Adaptive Framework for IoT

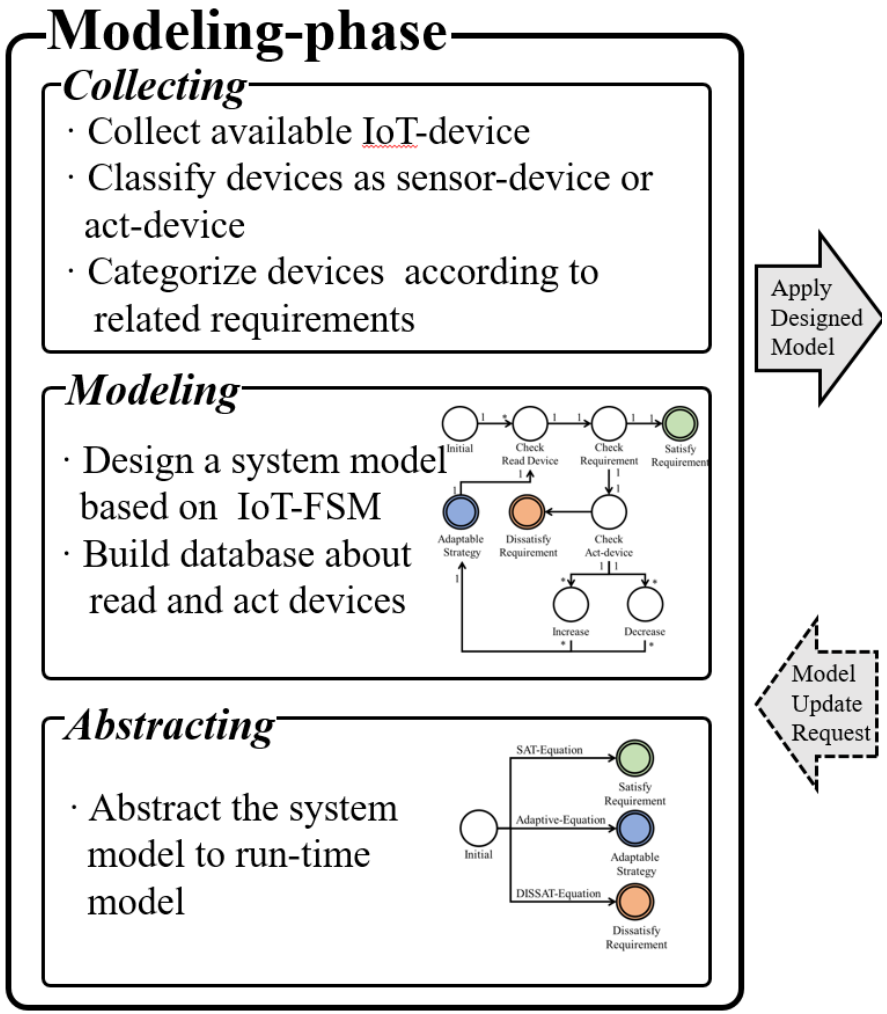
❖ Relationship between RINGA and RINGA-IoT



# RINGA-IoT: Self-Adaptive Framework for IoT

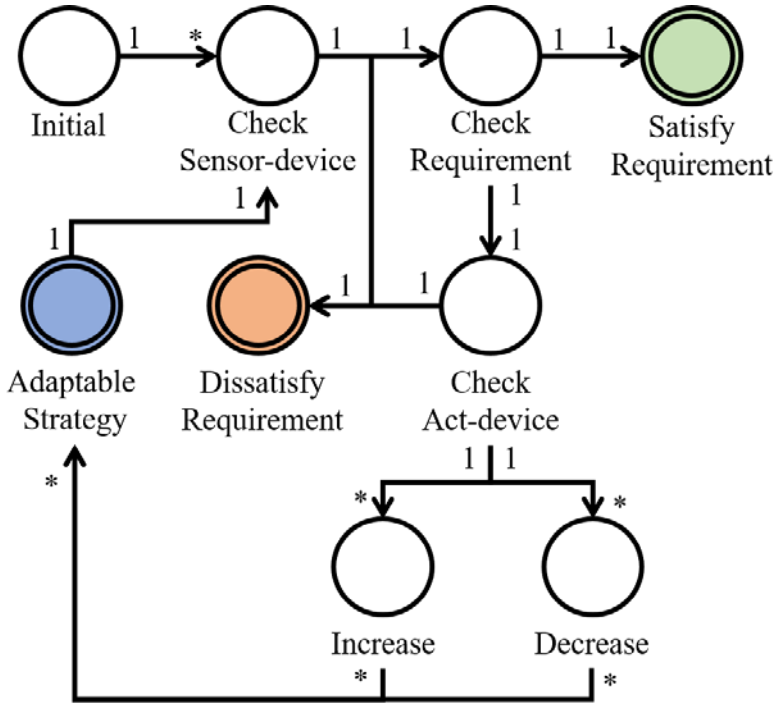


# RINGA-IoT: Self-Adaptive Framework for IoT




❖ IoT-FSM: Model for IoT

- FSM model based on SA-FSM for IoT



# RINGA-IoT: Self-Adaptive Framework for IoT

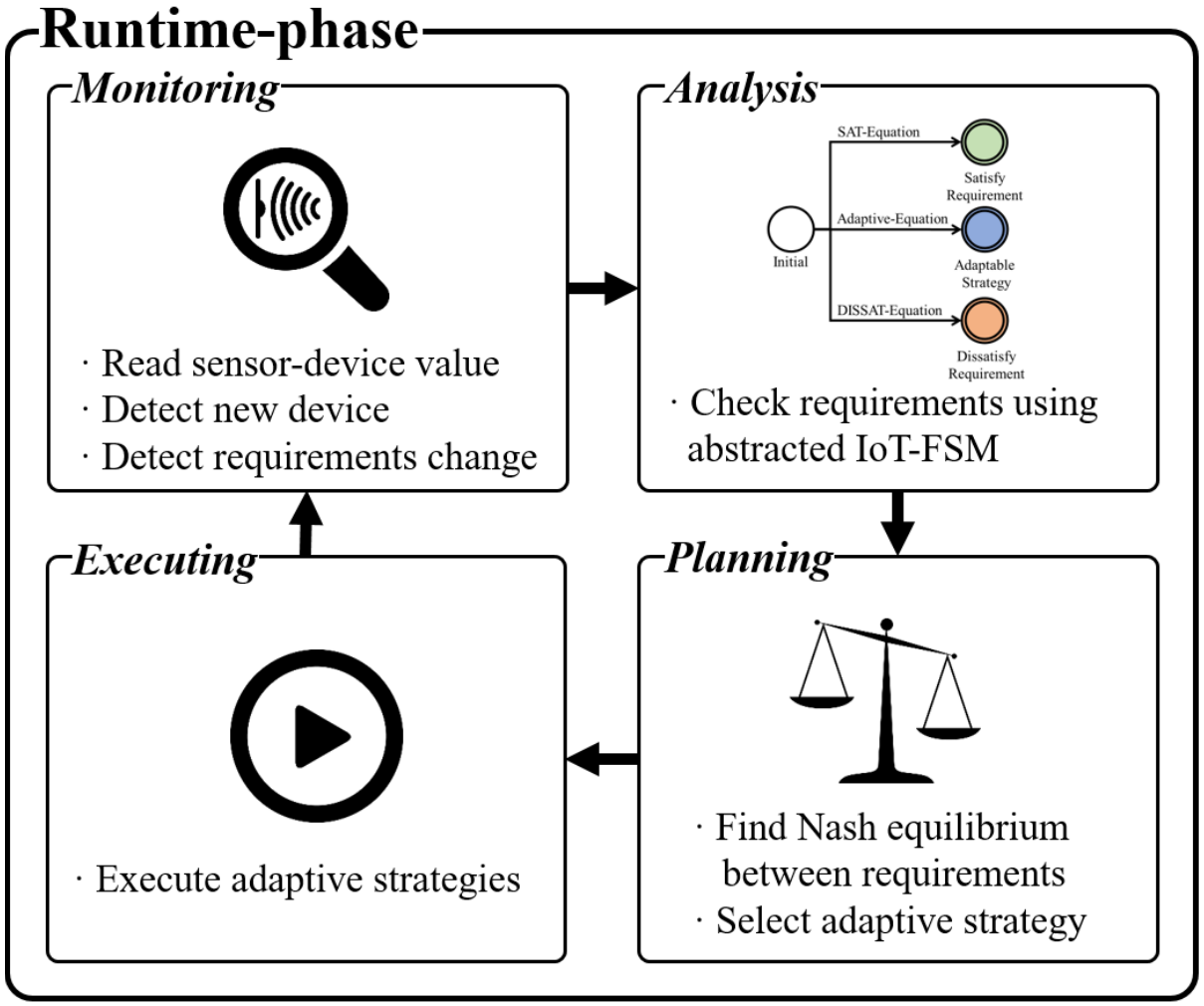
❖ MAPE-loop

- Monitoring-Analysis-Planning-Execution
- **NE-IoT**: Game Theory based Strategy Extraction
- Strategy score to select optimal solution at runtime

$$\text{Strategy Score} = \alpha\left\{\log\left(\frac{SR+1}{RR+1}+1\right)\right\} + \beta\left\{\log\left(\frac{1}{AD+1}+1\right)\right\}$$

※ $\alpha + \beta = 1$

*SR: the number of requirements that can be satisfied by the execution of a strategy*
*RR: the number of requirements that can be affected by the execution of a strategy*
*AD: The number of act devices that operate under an adaptive strategy*



# RINGA-IoT: S-A Software Framework for IoT

## ❖ NE-IoT: Game Theory based Strategy Extraction

- $S = S_1 \times S_2 \times \ldots \times S_n$ is the strategy set of profiles.
- Requirement $i \in \{1,\ldots,n\}$.
- $f(x)=\{f_1(x),\ldots, f_n(x)\}$ is the payoff function.
- A payoff function is evaluated at $x \in S$,
- $x_i$ is an act-device profile of player $i$.
- $x_{-i}$ is an act-device profile of the other players.
- Requirement $i$ operates act-device $x_i$, resulting in strategy profile $x=(x_1 \ldots x_n)$; then, requirement $i$ obtains payoff $f_i(x)$.
- $x^* \in S$ is a Nash equilibrium for IoT when $\forall i, x_i \in S_i$: $f_i(x_i^*, x_{-i}^*) \geq f_i(x_i, x_{-i}^*)$.
- $x^*$ can be an operation candidate at runtime.
- A strategy with the highest Nash equilibrium value among requirements is selected and implemented.

Requirement as a player of the game

Act-device as a strategy of the requirement

Nash equilibrium as a solution of the game
(=Solution satisfies all requirements)

# RINGA-IoT: S-A Software Framework for IoT

❖ Evaluating Strategies

$$\text{Strategy Score} = \alpha\left\{\log\left(\frac{SR+1}{RR+1}+1\right)\right\} + \beta\left\{\log\left(\frac{1}{AD+1}+1\right)\right\}$$

$$※\ \alpha+\beta=1$$

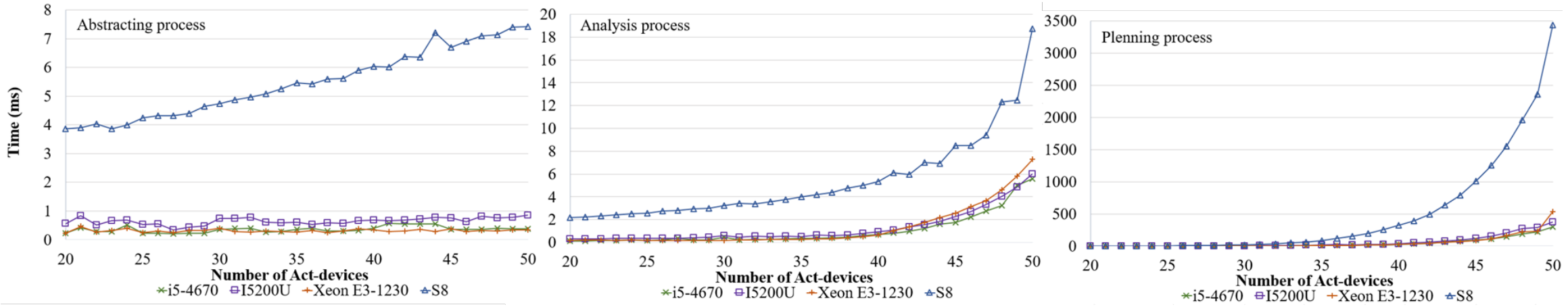
- SR: the number of requirements that can be satisfied by the execution of a strategy
- RR: the number of requirements that can be affected by the execution of a strategy
- AD: The number of act devices that operate under an adaptive strategy

**Maximum satisfied requirements, Minimum affected requirements, Minimum act-devices**

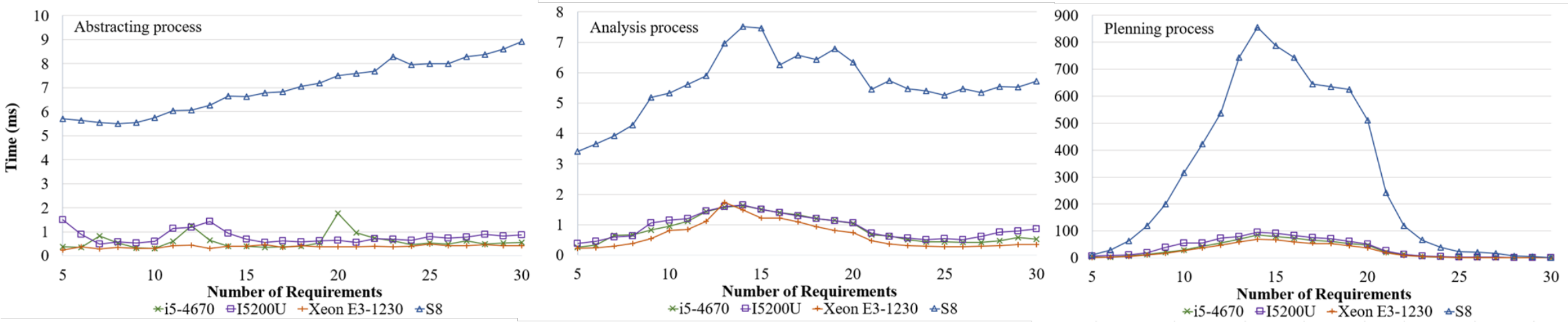
# RINGA-IoT: Self-Adaptive Framework for IoT

❖ MAPE-loop Evaluation (e.g., runtime performance evaluation)

- Fixed requirement(10) with increasing act-devices



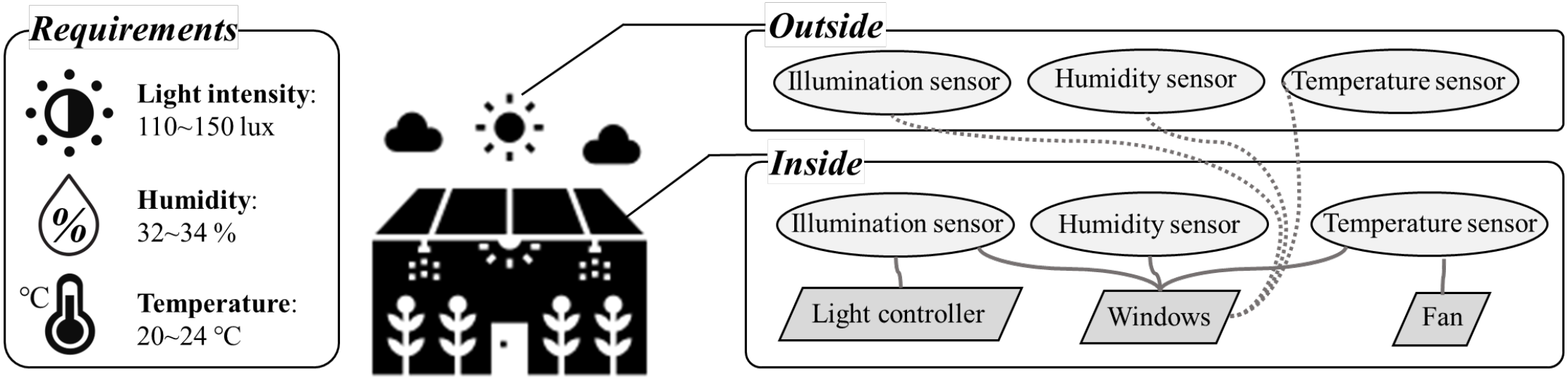

- Fixed act-device(40) with increasing requirements



# RINGA-IoT: Self-Adaptive Framework for IoT

❖ Proof of concept with IoT-based smart greenhouse scenario

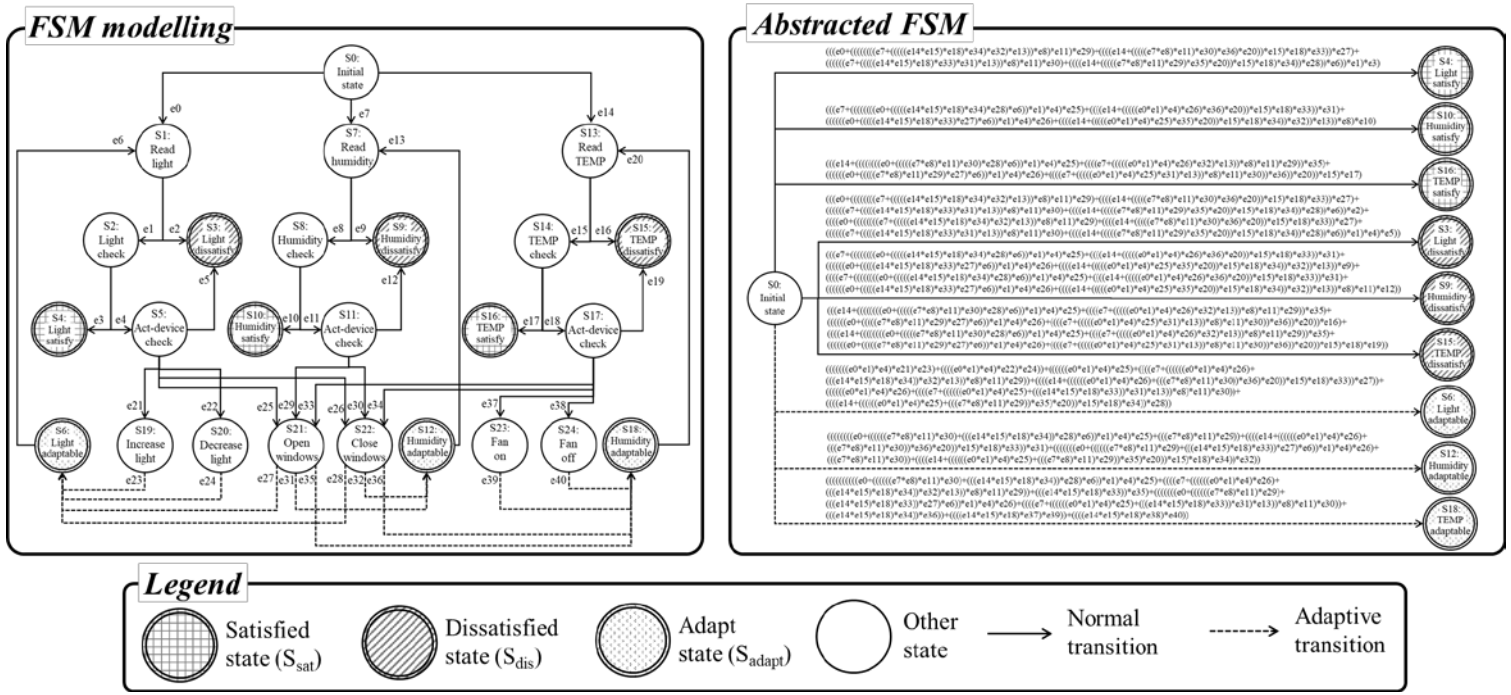
- Requirements: Light, Humidity, and Temperature
- Actuators: Light controller, Windows, and Fan
- Three scenarios with different situations



# RINGA-IoT: Self-Adaptive Framework for IoT

❖ Proof of concept with IoT-based smart greenhouse scenario (cont.)

- Models and abstracted model based on proposed framework
- The most optimal solution with game-theory based strategy extraction




< Modeling Results >

**Payoff matrix with external environment #1**

| | | Light | | | | |
|---|---|---|---|---|---|---|
| | | Increase (light) | Decrease (light) | Open (windows) | Close (windows) | |
| Humidity | Open (windows) | (1, 1, 1) ***SS = 0.458*** | (1, 0, 1) | (1, 1, 1) ***SS = 0.49*** | Operation confliction | On (fan) |
| | | (1, 1, -1) | (1, 0, -1) | (1, 1, -1) | Operation confliction | Off (fan) |
| | | (1, 1, -1) | (1, 0, 1-) | (1, 1, -1) | Operation confliction | Open (windows) |
| | | Operation confliction | Operation confliction | Operation confliction | Operation confliction | Close (windows) |
| | Close (windows) | (0, 1, 1) | (0, -1, 1) | Operation confliction | (0, 0, 1) | On (fan) |
| | | (0, 1, 0) | (0, -1, 0) | Operation confliction | (0, 0, 0) | Off (fan) |
| | | Operation confliction | Operation confliction | Operation confliction | Operation confliction | Open (windows) |
| | | (0, 1, 0) | (0, -1, 0) | Operation confliction | (0, 0, 0) | Close (windows) |

Temperature

<Strategy extraction example >

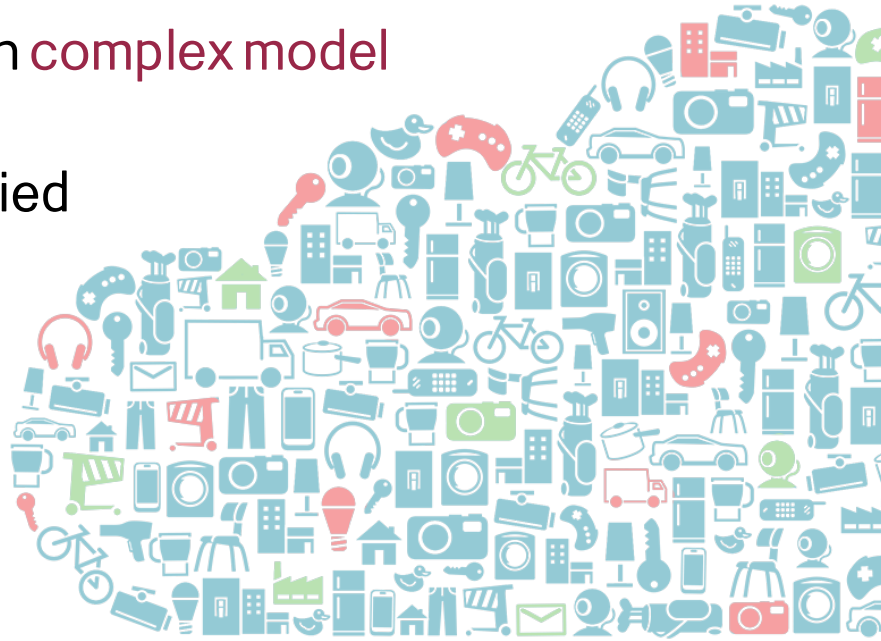
# Cache-based Model Abstraction

IEEE Internet of Things journal (2020)

# Objective

❖ Objective: Reduce Abstraction & Verification Time in RINGA

- IoT devices has low computing power
- RINGA required time to abstraction and verification time with complex model
- Abstraction and verification time have to be reduced to applied in low computing devices

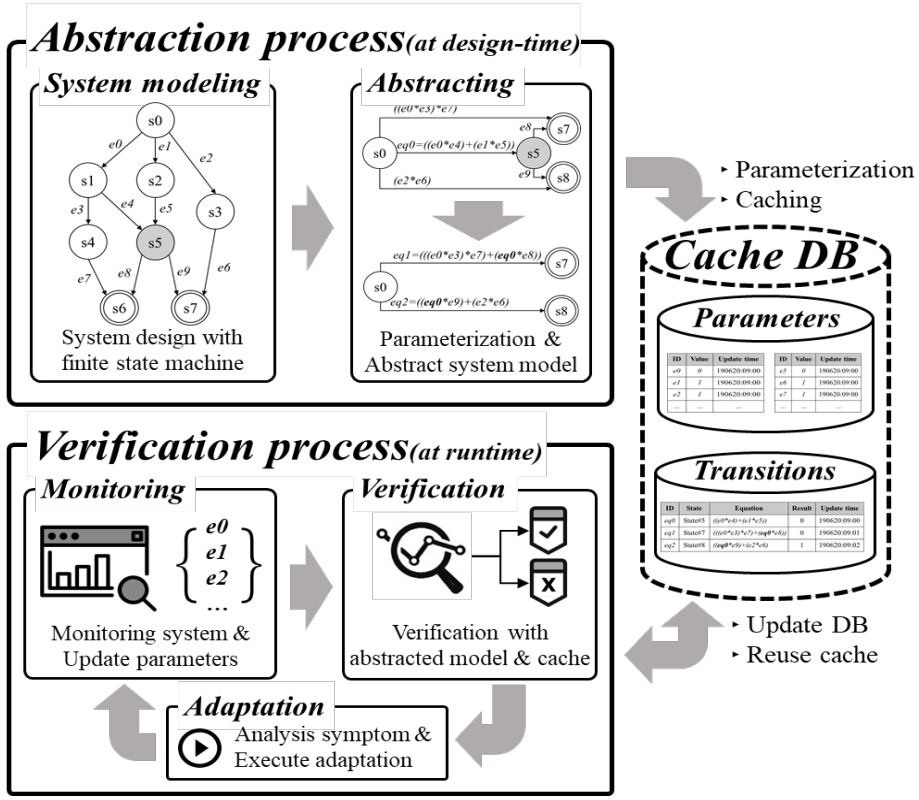


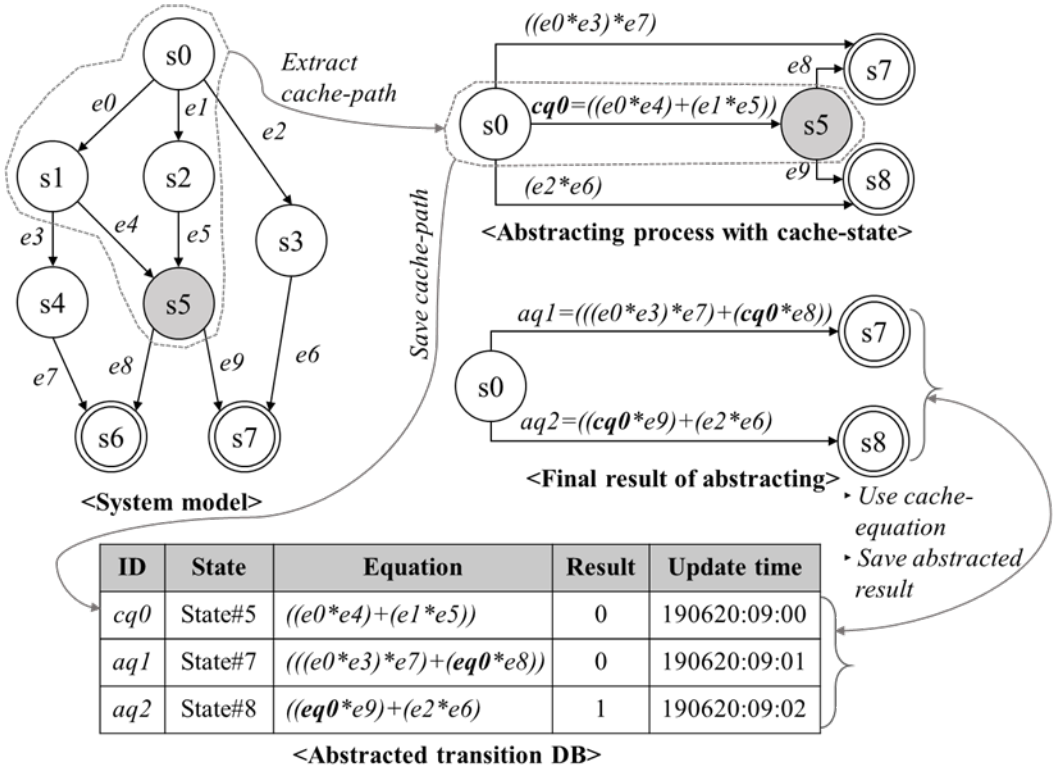
**A cached based abstraction and verification methods is proposed to apply RINGA in complex IoT environment**

# Enhanced RINGA Framework with Cache-Mechanism

## ❖ Demand & Objective

- Limitation of model checking in RINGA
  - ✓ RINGA required performance improvement to be applied with complex IoT environment
  - ✓ Verification for low computing power devices
- Caching mechanism is proposed to enhance RINGA



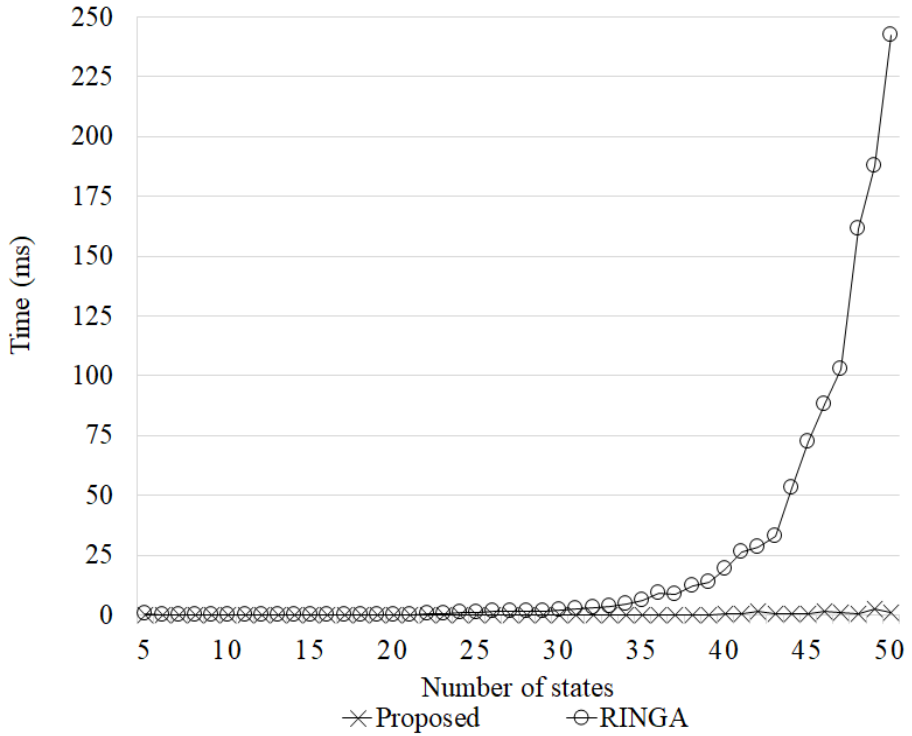



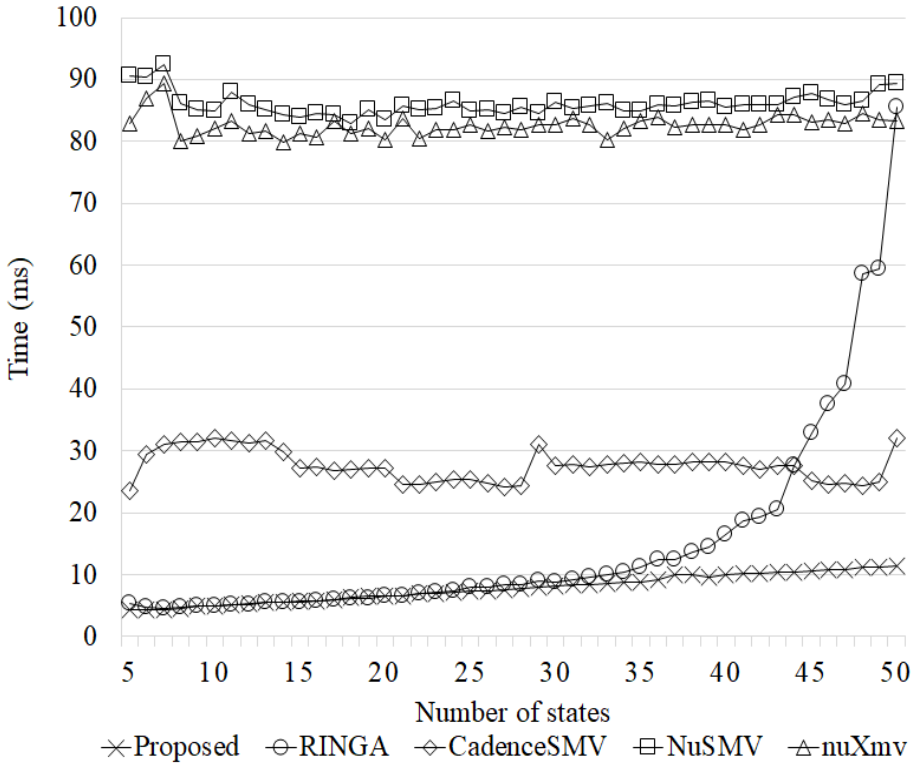

| ID | State | Equation | Result | Update time |
|---|---|---|---|---|
| cq0 | State#5 | ((e0*e4)+(e1*e5)) | 0 | 190620:09:00 |
| aq1 | State#7 | (((e0*e3)*e7)+(eq0*e8)) | 0 | 190620:09:01 |
| aq2 | State#8 | ((eq0*e9)+(e2*e6) | 1 | 190620:09:02 |

<Abstracted transition DB>

# Enhanced RINGA Framework with Cache-Mechanism

- Results and Status
  - Caching mechanism significantly improves performance of RINGA



# Development process of S-A Research

❖ Development process of self-adaptive software research

**RINGA**
*for Self-Adaptive*

SAC 2017

2018 Information and software technology

**RINGA-IoT**
*for Internet of Things*

TENCON 2018

2019 Sensors

**RINGA-ML**
*for Internet of Things*

**Game Theory based Strategy Extraction**
*for Self-Adaptive*

2019 Ambient intelligence & humanized computing

**RINGA**
with Cache Mechanism
*for Self-Adaptive*

IEEE Internet of Things journal

**RINGA**
with different architectures
*for Self-Adaptive*

# Thank You

kongjjagae@cbnu.ac.kr